ELAN Rev B3 Errata / Anomalies / Design Changes
Revised 07/17/95

Errata numbers preceded with “E” are carried over from the ELAN Rev. A, those preceded with “EB” are new to the Rev B3.

ERRATA # E36 — *DMAC Powers Up Disabled*

REVISIONS EFFECTED:	B_3 effected
VERIFIED:	
ERRATA DESCRIPTION:	In the 8237 logic, the DMA controller enable bit is not initialized correctly during reset. This is incompatible with the 8237 specification. This should not cause any customer problems, the DMA controllers can be initialized correctly during the chip set initialization.
SILICON SOLUTION:	None.
SYSTEM SYMPTOM:	None.
HARDWARE/SOFTWARE WORK AROUND:	In BIOS during system hardware initialization, the DMA controller enable bits should be set (enabled).

#EB4 — *Asymmetric DRAMs in non-enhanced page mode*

REVISIONS EFFECTED:	B_3 effected
VERIFIED:	
ERRATA DESCRIPTION:	16Mb asymmetric DRAMs used in a 2-bank configuration in non-enhanced page mode does not work.
SILICON SOLUTION:	None planned at this time
SYSTEM SYMPTOM:	DRAM failures
HARDWARE/SOFTWARE WORK AROUND:	If Index B4H bit [7] is set, and Index B1H bits [7:6] = '11', then Index 66H bits [1:0] must = '11'.

#EB12 *CPUCLK Idle at low-speed errata*

REVISIONS EFFECTED:	B_3 effected
VERIFIED:	
ERRATA DESCRIPTION:	This is a 33MHz and 25mhz issue only. This Errata was present in all Rev A parts also. Setting bit 2 of Index 6BH will cause a timing error in the clock-switching logic of the ELAN. This bit is used to force the CPUCLK to low speed, 9MHz, when the CPU is not performing a bus cycle.
SILICON SOLUTION:	None
SYSTEM SYMPTOM:	In general, the internal version of SYSCLK which keeps the phase of the 2x CPUCLK gets out of sync with CPUCLK, i.e. SYSCLK will rise going into PHI2 of CPUCLK instead of PHI1. This results in a wide variety problems: REFRESH pulse widths, incorrect ISA cycles, etc. Specifically, we observed that a 16-bit write to a 2 wait-state, internal IO decode resulted in XIOWR being asserted for the even address and not for the odd.
HARDWARE/SOFTWARE WORK AROUND:	1) Run at 20Mhz. or 2) Do not set bit 2 of Index 6BH (Idle Low Speed)

#EB14 *0WS# Timings Possibly Out of Specification*

REVISIONS EFFECTED:	B_3 effected
VERIFIED:	
ERRATA DESCRIPTION:	Elan is possibly out of spec for response timing requirements from falling edge of command during a 0 wait state cycle. ELAN needs a 20 nsec response or faster.
SILICON SOLUTION:	None.
SYSTEM SYMPTOM:	Invalid ISA cycles.
HARDWARE/SOFTWARE WORK AROUND:	Return 0WS# signal faster.

#EB 18 *Internal DRAM/Linear ROMDOS address conflicts*

REVISIONS EFFECTED:

B_3
effected

VERIFIED:

ERRATA DESCRIPTION:

When enabling the linear ROMDOS decodes, there must not be DRAM addressing conflicts with the programmed linear ROMDOS range. For example, if there is 8MB of DRAM in the system, the linear ROMDOS address range must not be programmed to decode below the 8MB boundary - Index B8h[3:0] must be greater than 7.
If the user wants to disable a portion of the DRAM to use as linear ROMDOS, the MMS Range logic in Index 6Fh[7:4] should be programmed BEFORE enabling the linear ROMDOS range to overlap existing DRAM. In the above example, if the user wanted 12MB of linear ROMDOS, Index 6Fh[7:4] should be set to '0100' before writing a '1100' to Index B8h[3:0].

SILICON SOLUTION:

None

SYSTEM SYMPTOM:

If the ELAN is configured such that the DRAM and linear ROMDOS configurations overlap, the ELAN will issue incorrect DRAM addresses on MA[10:0] for DRAM cycles that are also in the linear ROMDOS range.

HARDWARE/SOFTWARE WORK AROUND:

Do not configure the ELAN for linear ROMDOS/DRAM conflicts.
or
Always use the MMS range logic in Index 6Fh[7:4] to prevent linear ROMDOS/DRAM conflicts.

#EB 26 *PCMCIA Card Enables during SMI/NMI handlers in SUSPEND mode*

REVISIONS EFFECTED:

B_3
effected

VERIFIED:

ERRATA DESCRIPTION: During a temporary-on SMI/NMI handler when the PMU is in the SUSPEND or OFF PMU state, the PCMCIA Card Enables: (XMCEL_A, XMCEH_A, XMCEL_B, and XMCEH_B) are driven to '0's. This will cause the PCMCIA cards to drive data onto the SD bus during read cycles.

SILICON SOLUTION: Remove logic that forces PCMCIA Card enables low during SUSPEND or OFF modes.

SYSTEM SYMPTOM: The PCMCIA Card Enables will be driven low during temporary-on SMI/NMI handlers when the PMU is in SUSPEND or OFF mode. Unless the PCMCIA sockets' data drivers are disabled, this will cause data contention on the SD bus between the two PCMCIA sockets and any other decoded device during an ISA bus read. In addition, all write cycles that go to the ISA bus will cause data to be written into both PCMCIA sockets.

HARDWARE/ SOFTWARE WORK AROUND:

If the user does NOT desire to use temporary-on SMI/NMI handlers during SUSPEND or OFF mode than the user can disable temporary-on SMI/NMI handlers during SUSPEND and OFF modes. The control for this function is in Index 82h.

Alternate Work Around # 1:

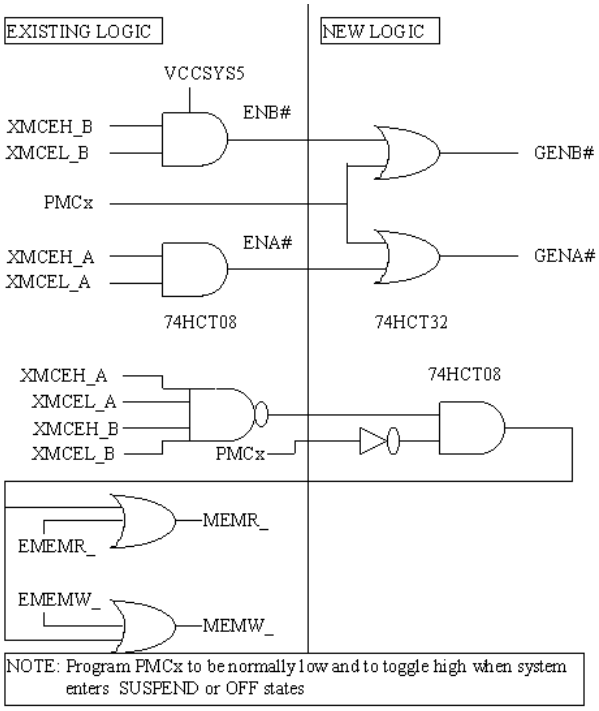
Use SLEEP as lowest level PMU mode, instead of SUSPEND or OFF.

Alternate Work Around # 2:

If the user desires to use temporary-on SMI/NMI handlers during SUSPEND or OFF mode AND the PCMCIA sockets will drive data onto the SD bus (i.e. their data bus drivers have not been powered off or disabled), the following H/W solution may be implemented:

Gate the PCMCIA Card Enables with a PMC line that is programmed to indicate that the PMU state SUSPEND has been reached. PMC Suspend state control functions are in registers 81h, ABh, and ACh. The gating function should disable the ELAN's PCMCIA Card Enables from causing the sockets to drive data onto the SD bus. In addition gate the inverted PMC line (programmed for SUSPEND) with the existing MEMR /MEMW lines which currently are gated off when any XMCEx line is asserted. The new logic will keep MEMR/MEMW asserted while in a temp SMI/NMI handler during SUSPEND or OFF mode. See pictorial below.

Note: this fix will disable PCMCIA accesses in SUSPEND and OFF modes.



#EB 27 *ISA Bus Conflicts during Fast ROM accesses*

REVISIONS EFFECTED:

B_3
effected

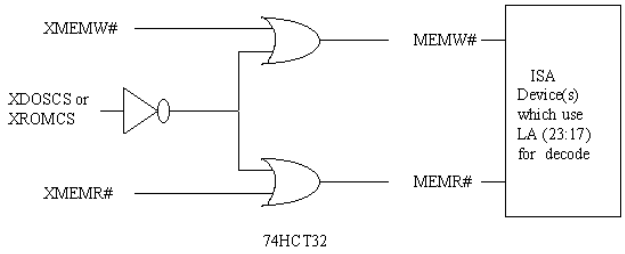
VERIFIED:

ERRATA DESCRIPTION: When the ELAN Rev. B is configured in the FULLISA pin-mode and ROM accesses are configured to run at the Fast CPU clock rate, the BALE (LVDD pin) output will not be asserted for FAST ROM accesses. If the last ISA cycle that asserted BALE is decoded by an ISA device that uses LA(23:17) in its decode, that device will conflict with FAST ROM accesses until BALE is reasserted.

SILICON SOLUTION: Assert BALE during FAST ROM accesses.

SYSTEM SYMPTOM: ISA bus conflicts will be seen during FAST ROM cycles after an ISA device that uses LA(23:17) has been accessed.

HARDWARE/SOFTWARE WORK AROUND: Externally 'OR' the ISA memory commands (XMEMR and XMEMW) with the inverted version of the ROM chip select(s) (XROMCS and/or XDOSCS) that are enabled to run at the Fast CPU clock rate. The ouputs of the 'OR' gates would then be routed to all ISA devices that use LA(23:17) to decode accesses. In addition, the XROMCS and XDOSCS outputs that will be used for FAST ROM accesses should be configured as address decodes and not internally qualified with the memory commands. This is programmable via Index B3H bit 2 (XROMCS) and Index B8H bit 4 (XDOSCS). See pictorial below.



#EB 28 *No Coprocessor Support*

REVISIONS EFFECTED:	**B_3 effected**
VERIFIED:	
ERRATA DESCRIPTION:	The Coprocessor will no longer be supported - there has been no demand from the field for this feature.
SILICON SOLUTION:	N/A
SYSTEM SYMPTOM:	N/A
HARDWARE/SOFTWARE WORK AROUND:	Do not use the coprocessor.

#EB29 *ELAN GATEA20 Functionality Clarification*

REVISIONS EFFECTED:

B_3
effected

VERIFIED:

ERRATA DESCRIPTION:

In the ELAN designs, the GATEA20 functionality is achieved through pin 79 (A20GATE), Index 6Fh bit 0 (A20G), and in the Rev. B design Port 92h bit 1 (ALTA20G). Additionally, Index 6Bh bit 0 (A20SMI) controls this function while the CPU is in the SMI handler state. The GATEA20 logic behaves as follows: If A20GATE, A20G, and ALTA20G are all low, and the CPU is not in SMI mode or A20SMI is low, CPUA20 will be gated low. Conversely, if A20GATE, A20G, or ALTA20G is high, or the CPU is in SMI mode and A20SMI is high, CPU20 will be ungated.

Pin 79 A20GATE	Index 6Fh[0] A20G	Port 92h[1] ALTA20G	CPU in SMI Mode	Index 6Bh[0] A20SMI	A20
0	0	0	No	x	0
0	0	0	Yes	0	0
1	x	x	x	x	CPUA20
x	1	x	x	x	CPUA20
x	x	1	x	x	CPUA20
x	x	x	Yes	1	CPUA20

A20G, ALTA20G, and A20SMI are initialized to 0's during an ELAN reset. The A20GATE pin has an internal pullup resistor. If the A20GATE pin is left unconnected, i.e. there is no 8042 in the system, the user will not be able to gate CPUA20 low via the internal registers in non-SMI mode. Tying the A20GATE pin to ground is generally not an acceptable solution since the internal A20GATE control bits power up low and the boot-vector fetch from FFFFF0h will not assert ROMCS#.

SILICON SOLUTION:

Change the internal control bits to initialize to 1's during an ELAN reset. This solution would require AT standard Software mods to clear these bits in systems that use external control for the A20GATE pin.

SYSTEM SYMPTOM:

Systems that don't have external control of the A20GATE pin, i.e an 8042, will not boot if the A20GATE pin is grounded and the boot- vector is located in a device that uses ROMCS#. If the A20GATE pins left unconnected, the user will not be able to gate CPUA20 low.

HARDWARE/SOFTWARE WORK AROUND:

Note: for the following solutions the user must insure that the A20GATE pin is driven high or one of the internal GATEA20 control bits is set before a CPU reset is performed. Otherwise, the ELAN will not assert ROMCS# in response to the boot-vector fetch.

SOLUTION #1:
Tie the A20GATE pin to ELAN PGP(0) or PGP(1). The appropriate PGP signal would then need to be configured as an output (set bit 6 of Index 70h for PGP(0) or bit 2 of Index 74H for PGP(1)), and configured to use direct control mode (clear bits 1:0 of Index 91h for PGP(0) or bits 3:2 of Index 91h for PGP(1)). The output state of the PGP signal would then be determined by Index 89h bit 7 for PGP(0) or Index 9Ch for PGP(1). The PGP signal could then be used to directly control the GATEA20 functionality, or the PGP output could be cleared and control of the GATEA20 function could be achieved via an internal control bit (A20G, etc.).

- OR -

SOLUTION #2:
Tie the A20GATE pin to the ELAN PMC(3) output. PMC(3) is controlled via Index ABh bits 3:0. PMC(3) could then be used to directly control the GATEA20 functionality, or the PMC(3) output could be cleared and control of the GATEA20 function could be achieved via an internal control bit (A20G, etc.).

- OR -

SOLUTION #3:
Add external control of the A20GATE pin. This could be an external flip-flop that is set by the ELAN's RSTDRV output and is cleared as a function of an address decode. The flip-flop could be used to directly perform the GATEA20 function, or the external flip-flop could be cleared, and control of the GATEA20 function could be achieved via an internal control bit (A20G, etc.).

#EB 30 *Potential Additional Current Consumption in uPower Mode*

REVISIONS EFFECTED:

B_3
effected

VERIFIED:

ERRATA DESCRIPTION: When the ELAN Rev. B enters MicroPower Mode, the internal logic that controls DRAM refresh is not disabled. This logic may consume as much as 60uA (a total of 75uA was observed on VCC_CORE) of current while in MicroPower Mode.

SILICON SOLUTION: Change the functionality of Index B3h bit 2 as follows:

When cleared (default state), the DRAM refresh logic will be disabled in the memory controller when MicroPower Mode is entered. The ELAN RAS and CAS signals will be forced low for the duration of MicroPower Mode. The sequencing of the RAS and CAS signals will not be guaranteed to maintain the DRAM contents.

When set, the ELAN will function as it currently does: the DRAM refresh logic will remain enabled during MicroPower Mode and the DRAM contents can be maintained during MicroPower Mode.

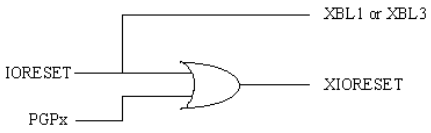
SYSTEM SYMPTOM: Additional current consumption on VCC_CORE during MicroPower Mode.

HARDWARE/SOFTWARE WORK AROUND:

Setting bit 3 of Index B3h, enable self-refresh DRAM Mode, limits the core current consumption to 16uA in MicroPower Mode.

For systems that use DRAMs that support self-refresh mode, no system changes need to be made. Also, systems that want to maintain the contents of DRAM during MicroPower Mode using CAS-before-RAS refresh will not need to make system changes since the power consumption will be dominated by the DRAMs and the I/O switching.

Systems that do not use self-refreshing DRAMs and do not want to preserve the contents of DRAM during MicroPower Mode must detect that the system is going to enter MicroPower Mode and set bit 3 of Index B3h if the power consumption mentioned above is unacceptable. Detecting MicroPower Mode may be accomplished by using a battery-low detection pin that the system is not using and another signal to gate XIORESET prior to the ELAN pin. XBL1 and XBL3 generally have the same functionality, so the system may be able to use one of these pins to detect a MicroPower Mode request. Assuming that XBL3 will be used to detect MicroPower Mode requests, the system's IORESET signal would be routed to the XBL3 input of the ELAN as well as the input of an 'OR' gate. The other 'OR' gate input could be connected to an ELAN PGP signal. The output of the 'OR' gate would then be connected to the ELAN's XIORESET pin. When the system detected a MicroPower Mode request via the XBL3 function, S/W would set bit 3 of Index B3h, and would clear the PGP pin to allow transition into MicroPower Mode. Upon exiting MicroPower Mode, S/W would need to clear bit 3 of Index B3h. See picture below.



NOTE: NMI/SMI generated by XBL1/3 detects uPower mode. PGPx programmed in handler to halt/propagate IORESET. PGPx initially high.

#EB 31 *System Hangs caused by Index 8Fh bit 4*

REVISIONS EFFECTED:

B_3
effected

VERIFIED:

ERRATA DESCRIPTION:

When Index 8Fh bit 4, (D1MS), is set, the PMU will disable the CPUCLK for 1ms when entering High-Speed PLL Mode. However, the 1ms timer used for this function is disabled if ACIN is asserted (high), S/W ACIN (Index 70h bit 5) is set, or the High-Speed PLL Transition Timer (Index 83h) = 00h. If any of the above conditions are true when the PMU enters High-Speed PLL mode from Low-Speed PLL Mode and the High-Speed PLL is not shut off in Low-Speed PLL Mode (Index ADh bit 2 = '0'), the ELAN will lock-up with no CPUCLK.

In addition, if the PMU transitions from Low-Speed PLL Mode to High-Speed PLL Mode due to an activity (as opposed to an Index 88h write), Index 8Fh bit 4 is set, and Index ADh bit 2 = '0', then the CPUCLK will hang if the activity is not cleared before the first refresh after entering High-Speed PLL Mode.

SILICON SOLUTION: Disable bit 4 functionality.

SYSTEM SYMPTOM: The ELAN may lock-up when returning to the High-Speed PLL Mode from Low-Speed PLL Mode when Index 8Fh bit 4 is set.

HARDWARE/SOFTWARE WORK AROUND: Don't set bit 4 of Index 8Fh. Note: This has negligible system impact.

#EB 32 *SMI/NMI Generation From XBL2 Assertions*

REVISIONS EFFECTED:

B_3
effected

VERIFIED:

ERRATA DESCRIPTION:

The status bit that indicates that an XBL2 assertion has occurred (Index A5h bit 6) is cleared when the PMU enters SLEEP, SUSPEND, or OFF mode. If the CPU receives the XBL2 SMI/NMI prior to entering these PMU Modes and the PMU subsequently transitions to SLEEP, SUSPEND, or OFF Mode before Index A5h is read, the SMI/NMI source will be cleared.

SILICON SOLUTION: None

SYSTEM SYMPTOM: SMI/NMI events whose source is unidentifiable.

HARDWARE/SOFTWARE WORK AROUND: This solution assumes that Index 88h is not used to force PMU state transitions.

A temporary-on SMI/NMI handler for DOZE-to-SLEEP PMU state transitions should be enabled by setting bit 2 of Index 82H. The handler for this event would always be able to determine if an XBL2 assertion NMI/SMI was generated, preventing unidentifiable SMI/NMI events from being generated.

#EB 33 *Battery Management Issues*

REVISIONS EFFECTED:

B_3
effected

VERIFIED:

ERRATA DESCRIPTION:

The PMU does not mask activities in the lower PMU states (SLEEP, SUSPEND, and OFF) when the battery-low indicator pins XBL2 and/or XBL4 are asserted (low). If the PMU senses activity in these states, undesired power consumption may occur when the PLLs and CPUCLK are restarted.

The worst case is as follows: if no SMI/NMI events are generated on state transitions (Index 82h bits 1 - 4 are cleared), the PMU is in DOZE Mode, XBL2 is asserted (low), an SMI/NMI event occurs because bit 6 of Index 82h is set, the PMU enters SLEEP Mode because bit 6 of Index 74h is cleared, and the PMU senses activity via Indices 08h, 75h, or 76h, the PLLs will be started and the CPUCLK will continue to be gated off. This state will be maintained until ACIN is asserted or XBL4 and XBL2 are de-asserted.

SILICON SOLUTION: None

SYSTEM SYMPTOM: Additional power consumption in the ELAN during battery-low conditions.

HARDWARE/SOFTWARE WORK AROUND:

The following describes a method for preventing excess power consumption during battery low conditions. This solution would require the system XBL4 signal to be connected to an input that can be read by the CPU if a four-level battery management system was desired. Also, this method assumes that Index 88h is not used to force PMU state transitions and Index 74h bits 7 and 6 are cleared.

The user should enable a temporary-on SMI/NMI handler for PMU state transitions by setting at least one bit of Index 82h bits 1 or 2. The SMI/NMI handler for this event would poll the battery-low indicator bits for assertions. If a battery-low condition was detected, the user could prevent all wake-up activities, except ACIN assertions, by writing FFh to Indices 08h and 76h, and BFh to Index 75h. After writing to Index A5h to clear the SMI/NMI event, the SMI/NMI handler would poll for ACIN or XBL2 and XBL4 to detect the wake up condition. The write to Index A5h would cause the PMU to transition to SLEEP mode where the ELAN could only be awoken by ACIN assertions or asserting XBL2 and XBL4 and pushing the RESUME key. Upon waking up, the CPU would continue execution of the SMI/NMI handler. The activity masks could be returned to their previous states prior to exiting the SMI/NMI handler.

#EB34 *Refresh and DMA lost after IRQ cycle*

REVISIONS EFFECTED:

B_3
effected

VERIFIED:

ERRATA DESCRIPTION: Following an IRQ cycle (INT/ACK), the bus controller logic will not generate any further HOLD requests to the CPU. This not only stops refresh (which requires a HLDRQ/HLDACK with the CPU) but prevents any subsequent DMA activity until a DRAM write is performed.

SILICON SOLUTION: None.

SYSTEM SYMPTOM: Refresh stopped, no DMA activity.

HARDWARE/SOFTWARE WORK AROUND: This is not usually an issue for most applications since part of the IRQ process involves pushing flags, CS, and IP onto the stack which typically requires a DRAM write. Special diagnostic code may see this as an issue if the stack is not located in DRAM. In this case, the workaround is to perform a dummy DRAM write. If the dummy write is to an area that's been set up as shadow RAM, shadow RAM write protect must be disabled by clearing bit 7 of index 65h.

#EB35 *8254 input clock runs slow*

REVISIONS EFFECTED:

B_3
effected

VERIFIED:

ERRATA DESCRIPTION: The input clock to the on board 8254 is divided down from the low speed PLL frequency. This results in an input clock of 1.1892 MHz Vs. the AT compatible standard of 1.19318 Mhz.

SILICON SOLUTION: None.

SYSTEM SYMPTOM: The system timer tick runs approximately 0.33 % slower than on a standard AT. This results in a loss of about 4.75 minutes per 24 hour interval. Since the DOS clock gets reloaded from the RTC chip each time the system is reset or powered up, this is more of an issue for systems that are left running for long periods of time.

Although the slow input clock also affects timer 1 and timer 2, the effects of the off -spec input frequency are negligible on the refresh rate and speaker data timer functions controlled by these timer channels.

HARDWARE/SOFTWARE WORK AROUND: Timer 0 counter latch is normally loaded with 0FFFFh (64k) by BIOS. 1.19318 / 64k = 18.2 in an AT compatible system in order to generate 18.2 timer ticks per second. Since the 8254 runs slow, a value of less than 64k may be loaded into timer 0 to obtain the desired 18.2 ticks per second. Specifically, a value of 0FF3Ch may be loaded into the counter latch for timer 0 to obtain AT compatible timer tick performance.

#EB36 *Temp-Ons may allow DMA Txfr to Powered down Resource*

REVISIONS EFFECTED:	**B_3 effected**
VERIFIED:	
ERRATA DESCRIPTION:	If a DMA transfer is started (asserted by external device) while the system is in a clock OFF state with resources powered OFF, and a TEMP-ON handler is initiated - the DMA transfer will begin (if unmasked) and may attempt to access a powered down resource - which could result in numerous undesirable affects - bad data, corrupting a device, system hanging, etc.
SILICON SOLUTION:	None.
SYSTEM SYMPTOM:	External devices may cause numerous undesirable affects attempting a DMA access to a powered down resource (DRAM, PCMCIA, etc.).
HARDWARE/SOFTWARE WORK AROUND:	Program the system to generate an SMI/NMI when transitioning from clock on to clock off states, and mask OFF any DMA transfers which may attempt to access powered down resources. Wake-ups (in the SMI/NMI routine) can re-enable DMA txfrs after appropriate resources have been powered ON.

#EB37 *RESUME Event May Cause System Hang*

REVISIONS EFFECTED:

B_3
effected

VERIFIED:

ERRATA DESCRIPTION:

If the PMU receives a temporary-on event while the Low Speed PLL (LSPLL) is shut down and the PMU is in SLEEP, SUSPEND, or OFF mode, and a RESUME key press is received during the 256ms PLL startup interval (as defined by Index 8FH bits 2-0), the RESUME event will force the PLL startup timer to be reset causing the internal clocks to remain shut off. Since the internal clocks remain gated off, the system will hang. When this condition occurs, the system may be restarted by forcing XBL4 or XBL2 low while ACIN is low and the appropriate bit is set in Index 74H bits 6 or 7.

Applicable temporary-on events for this errata include: SLEEP to SUSPEND transition SMI/NMIs, SUSPEND to OFF transition SMI/NMIs, or SMI/NMIs caused by battery-low indicators XBL3 or XBL1.

SILICON SOLUTION: Qualify the PLL startup timer reset signal such that a RESUME event does not force a reset during temporary-on mode.

SYSTEM SYMPTOM: RESUME events generated by the SUS/RES key may cause system hangs when temporary-on SMI/NMIs are enabled in SLEEP, SUSPEND, or OFF modes.

HARDWARE/SOFTWARE WORK AROUND:

Disallow generation of SMI/NMIs on state transitions from SLEEP to SUSPEND or SUSPEND to OFF. Also, disallow XBL1 and XBL3 events from generating SMI/NMIs while the PMU is in SLEEP, SUSPEND, or OFF modes. This is accomplished by clearing bits 7, 5, 4, and 3 of Index 82H.

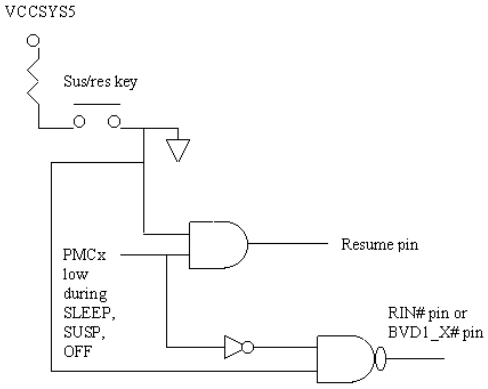
- or -

Leave the Low Speed and Video clocks enabled in SLEEP, SUSPEND and OFF modes by clearing bits 3 and 7 of register 81h (Power Control Reg 2).

- or -

Gate Off the SUS/RES switch signal from the Resume pin on the ELAN using a PMC or PGP signal, such that the RESUME function is disabled while the PMU is in SLEEP, SUSPEND, or OFF modes. Also Gate the SUS/RES switch signal to an unused wake-up pin, such as RIN#, BVD1_X# , etc, to effectively act as the Resume signal to wake-up the system.

Since gating off the SUS/RES pin eliminates the ability to generate a Resume SMI, the following SW implementation is suggested if this functionality is necessary. Enable a DOZE to SLEEP transition SMI /NMI (will catch SUSPEND action of SUS/RES switch, BL2 and BL4). Once in the SMI routine, read the NMI/SMI control reg A5h, if bit 2 is set (DOZE to SLEEP transition) power down appropriate devices, write 0 to reg A5h (allow transition to SLEEP), then jump into a SW loop polling on PMU mode/and or Status bits (polling will halt once SLEEP mode is reached and the CPUCLK is halted). When an activity, wake-up, SUS/RES key press (via RIN or BVD1_X), or Temp on event (BL1, BL3 , or PMU state transition) is received, the loop will resume, powered down devices can be re-enabled and the SMI/NMI handler exited when the polling loop detects FULL-ON mode. If the event warrants staying in a clock OFF mode - SLEEP, or transitioning to a lower PMU mode (SUSPEND or OFF) than clear NMI/SMI via reg A2H write, do appropriate tasks and remain in the SW loop. See pictorial below:



#EB38 *PCMCIA -REG signal not controllable on a per page basis*

REVISIONS EFFECTED: B_3 effected

VERIFIED:

ERRATA DESCRIPTION: The -REG signal on the PCMCIA interface is used to support two different memory regions that may be available on a PC Card: Common memory and Register memory (often called attribute memory). The Elan PCMCIA controller supports a single control for the -REG signal for each of the two slots that it supports. The PCMCIA 2.1 socket services API specification supports a SetPage function that allows software to control -REG on a "per page" basis as opposed to the "per slot" basis now supported by Elan.

Therefore, it is possible for conflicts to arise when an application intends to keep Common and Register memory active at the same time - by using multiple pages.

SILICON SOLUTION: None.

SYSTEM SYMPTOM: PCMCIA drivers and applications may not be compatible with the Elan PCMCIA controller. The Elan PCMCIA controller is not fully PCMCIA 2.1 compliant.

HARDWARE/SOFTWARE WORK AROUND: None.

#EB 39 *IRQ0 and PCMCIA Status Change SMIs may be missed*

REVISIONS EFFECTED: B_3 effected

VERIFIED:

ERRATA DESCRIPTION: If the PCMCIA Status Change or IRQ0 are configured to generate SMIs instead of interrupts and a PCMCIA Status change or IRQ0 occurs at the same time index A2 is written, the event or events will not generate an SMI and the events will not be recorded in the SMI status register Index 43.

Note: Writing to Index A2 is done during an SMI to clear the PMU's SMI request register.

SILICON SOLUTION: None.

SYSTEM SYMPTOM: In the case of IRQ0 being missed, the system may appear to hang because the SMI routine that would send an EOI to the interrupt controller will not get executed. This will cause all lower priority interrupts to remain masked off.

In the case of the PCMCIA Status Change not being recognized the system will not recognize the status change as having occurred with could cause various PCMCIA related errors depending upon the applications being executed.

HARDWARE/SOFTWARE WORK AROUND: The easiest solution is to implement both the PCMCIA Status Change and IRQ0 as interrupts and not SMIs.

- OR -

For the case of IRQ0: Mask off IRQ0 at the Programmable Interrupt Controller prior to writing Index A2, after writing Index A2, Unmask IRQ0.

For the case of PCMCIA status: Index A6 should be read after Index A2 is written to determine if any status change has occurred.

#EB 40 *Refreshes not disabled when using SRAM I/F*

REVISIONS EFFECTED:

B_3
effected

VERIFIED:

ERRATA DESCRIPTION: When the ELAN is configured to use SRAM for main memory (bit 0 of Index 70h is set), the CAS signals are asserted when the internal refresh signal is generated. For this SRAM configuration, the CAS signals are used for the SRAM Chip Select signals. Assertion of the CAS signals may cause the following problems:

1) If two banks of SRAM are used, and the Output Enables of the SRAMS are grounded, then there will be data driver contention on the D(15:0) pins during refreshes.
2) Higher system power consumption.

SILICON SOLUTION: Disable CAS assertions during refresh when the SRAM configuration is chosen.

SYSTEM SYMPTOM: Data driver contention on the D(15:0) signals when the ELAN system is configured to use two banks of SRAM for main memory. Also, additional system power consumption will be observed since the SRAM's Chip Selects will be periodically asserted.

HARDWARE/SOFTWARE WORK AROUND:

To eliminate contention on the D(15:0) signals one of the following should be implemented:
Only allow a single bank of SRAM.

- or -

Invert the ELAN's MWE# output signal, and connect the inverted version of MWE# to the Output Enables of the SRAMs.

To eliminate system power consumption created by refreshes, the following should be implemented:
Clear bit 2 of Index 48h to enable the MEMR# and MEMW# commands to be asserted during SRAM accesses. AND the MEMR# and MEMW# signals to generate an active low MEM_CMND signal. For each byte of SRAM used, OR the associated CAS signal with this MEM_CMND signal prior to connecting it to the CS pin of an SRAM. If this solution is implemented, the system designer should also qualify the Memory Command signals sent to any ISA device that does not fully decode the ISA address signals. See pictorial below: